Fig. 1

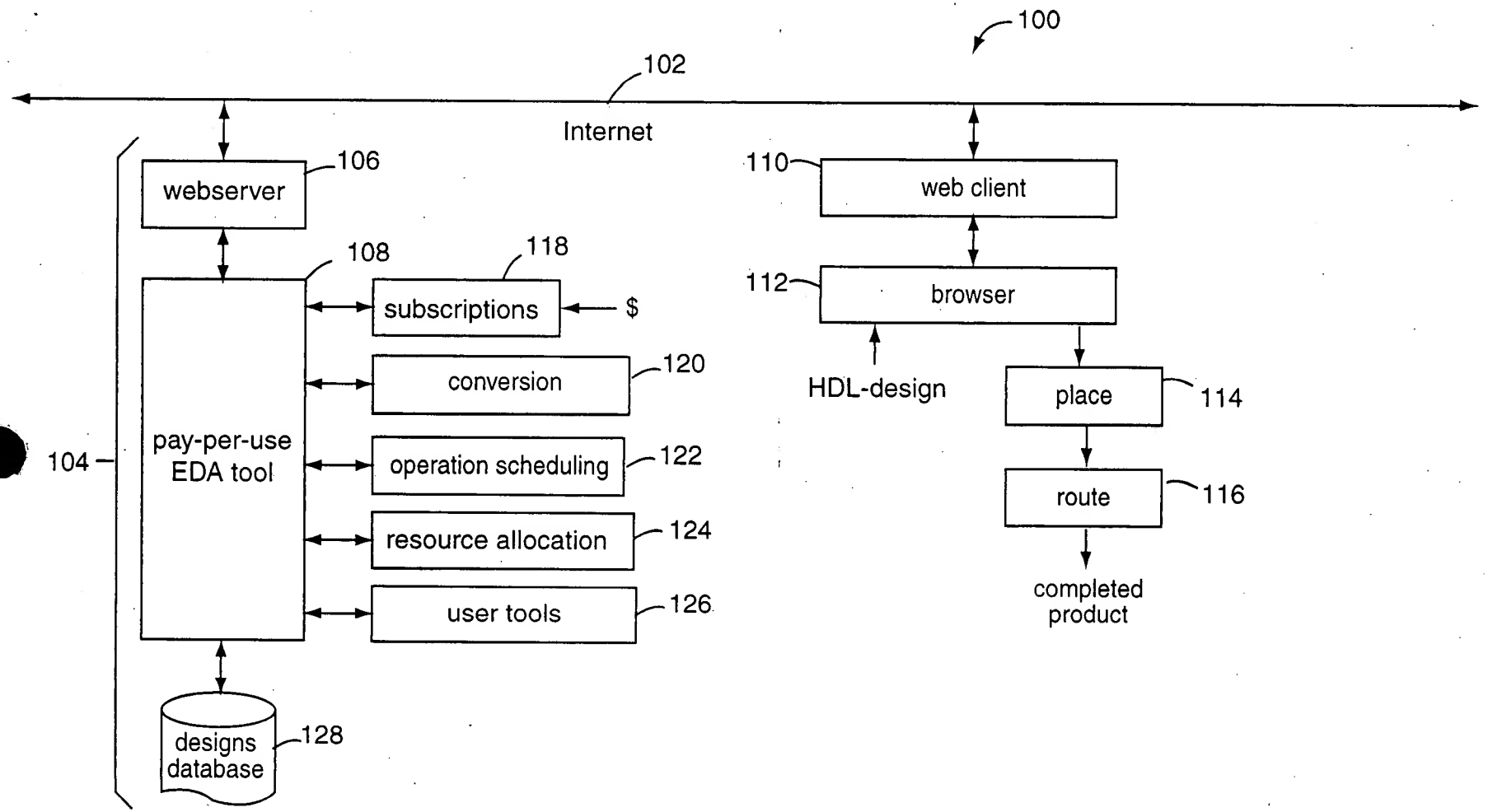


Fig. 2

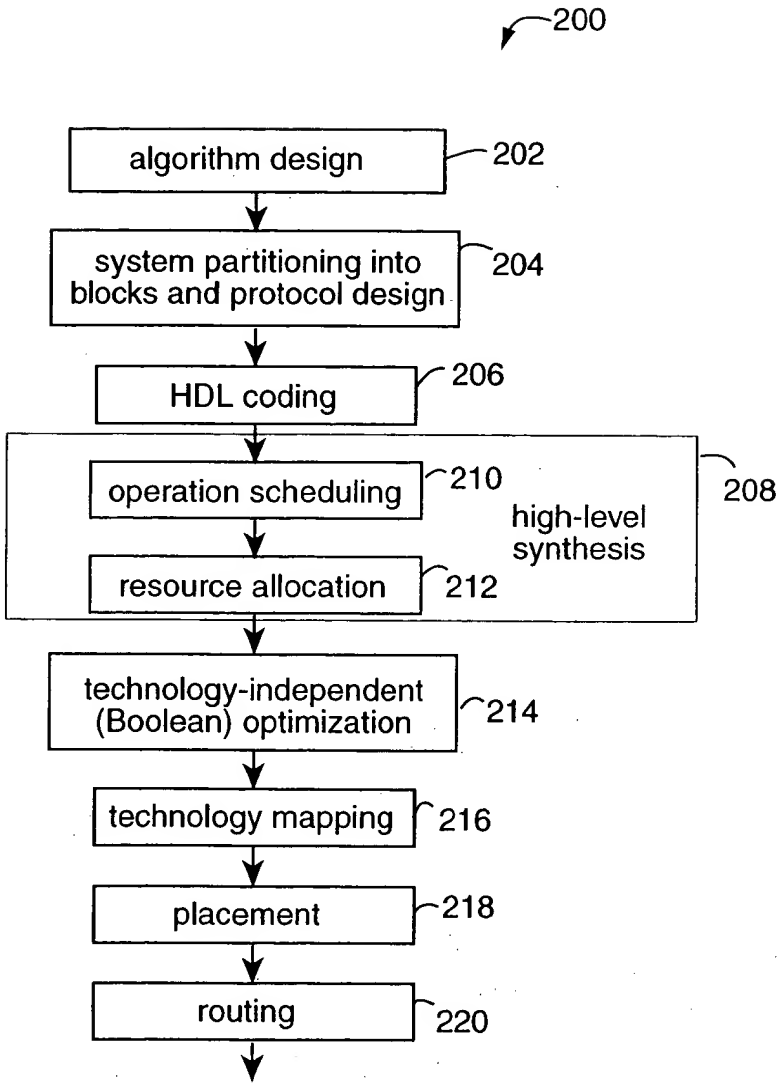

200
algorithm design
202
system partitioning into blocks and protocol design
204
HDL coding
206
operation scheduling
210
high-level synthesis
208
resource allocation
212
technology-independent (Boolean) optimization
214
technology mapping
216
placement
218
routing
220

Fig. 3

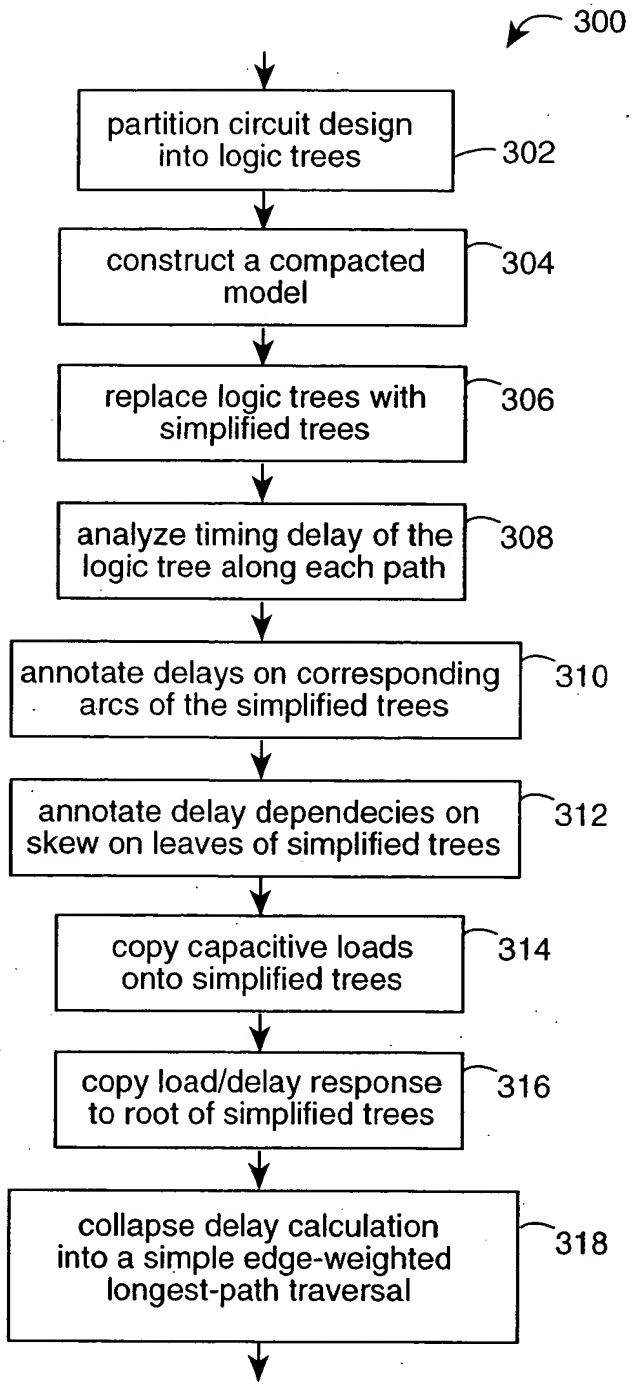

300
partition circuit design into logic trees
302
construct a compacted model
304
replace logic trees with simplified trees
306
analyze timing delay of the logic tree along each path
308
annotate delays on corresponding arcs of the simplified trees
310
annotate delay dependecies on skew on leaves of simplified trees
312
copy capacitive loads onto simplified trees
314
copy load/delay response to root of simplified trees
316
collapse delay calculation into a simple edge-weighted longest-path traversal
318

Fig. 4A

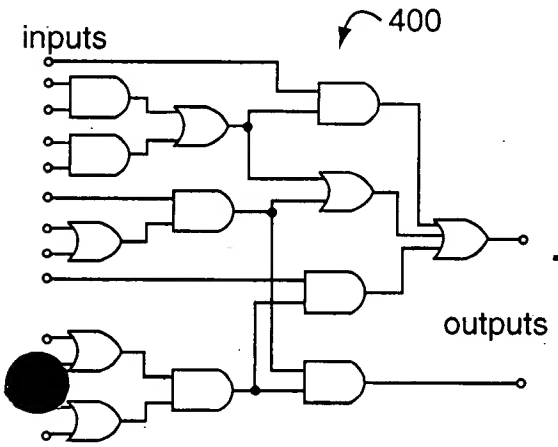


Fig. 4B

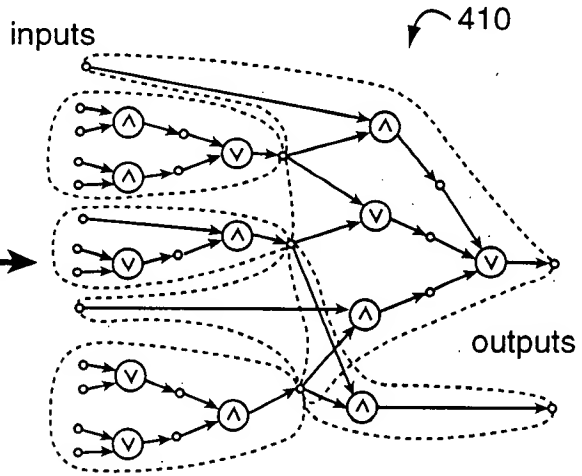


Fig. 4C

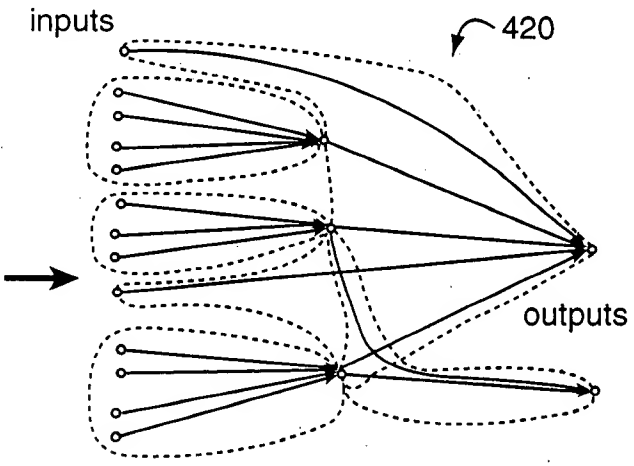
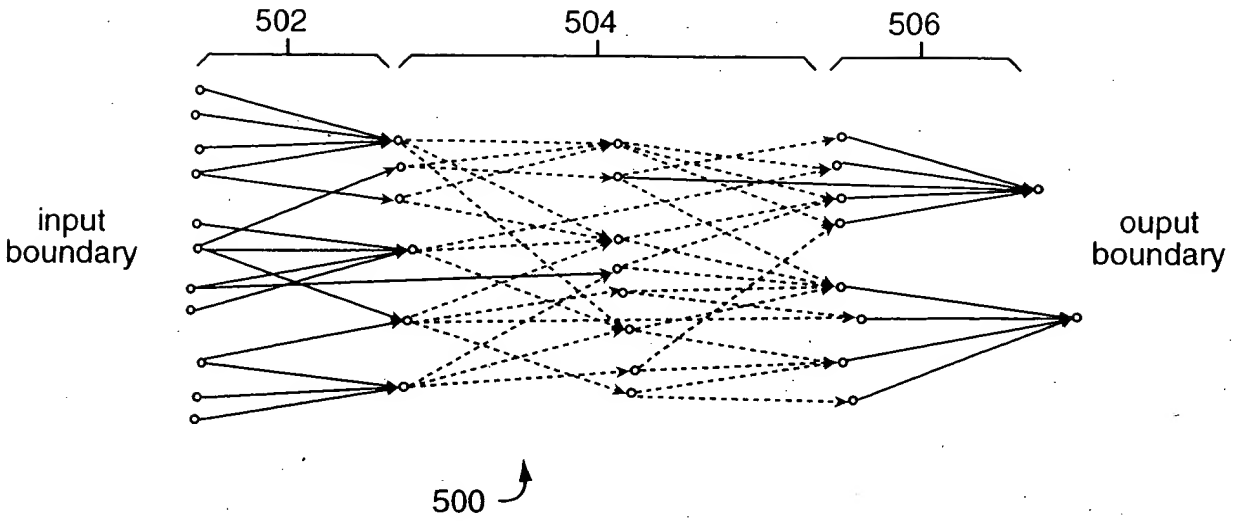


Fig. 5



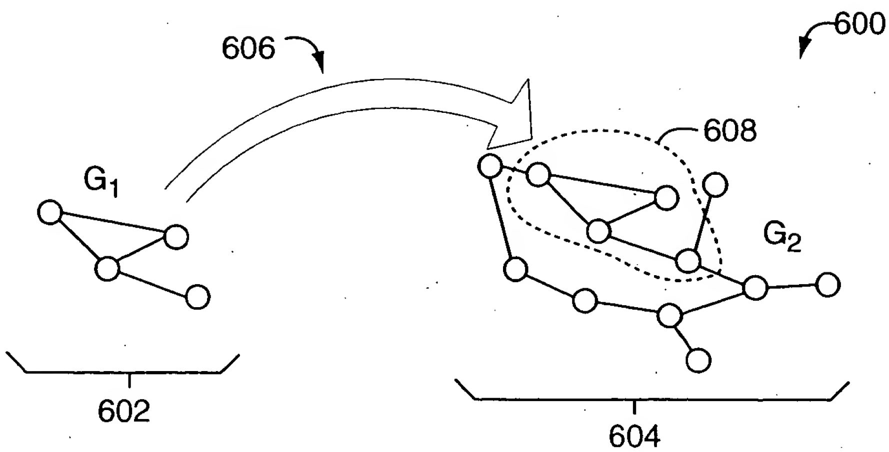
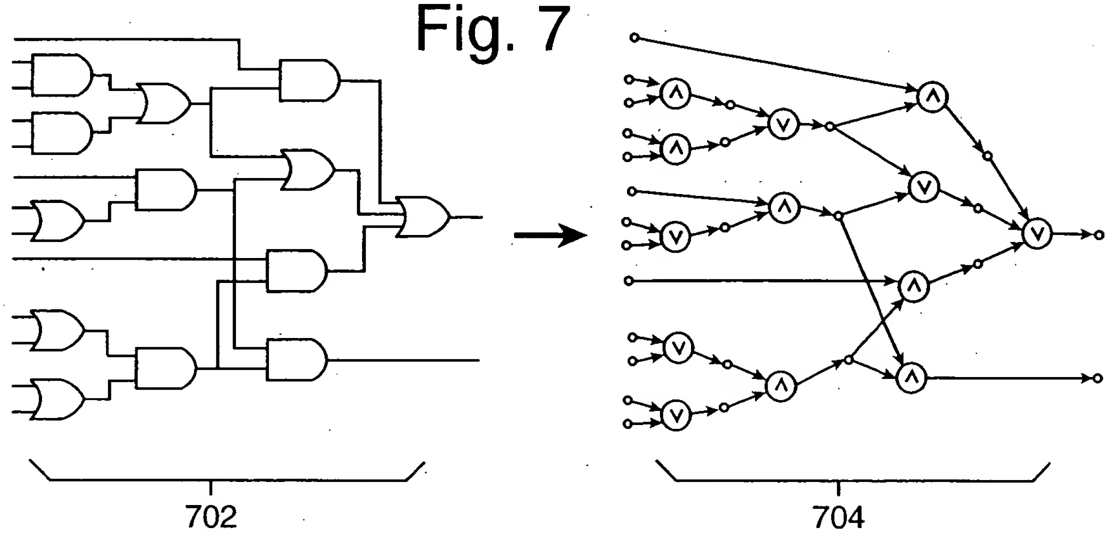


Fig. 6

Fig. 7



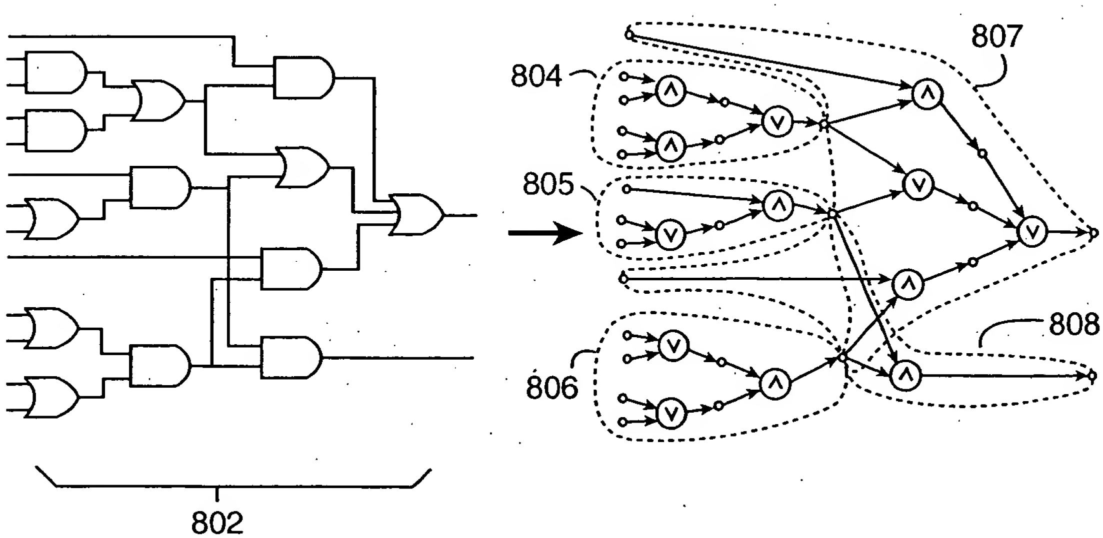


Fig. 8

Fig. 9

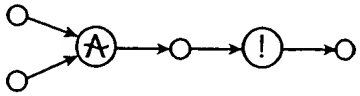
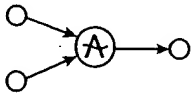
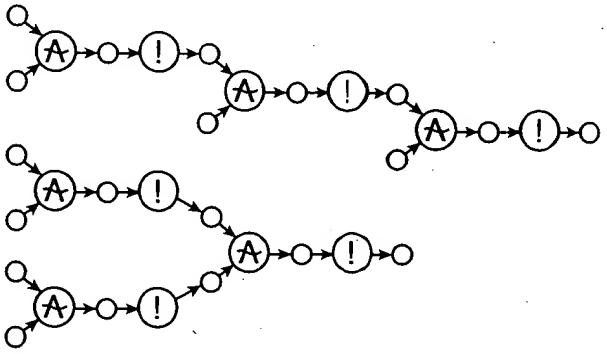
inv	! a	
and2	a b	
nand2	! (a b)	
and4	a b c d	

Fig. 10

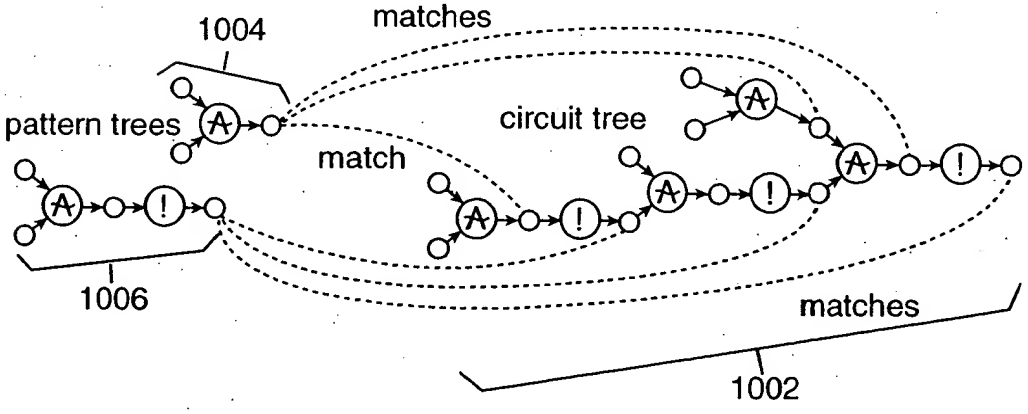


Fig. 11

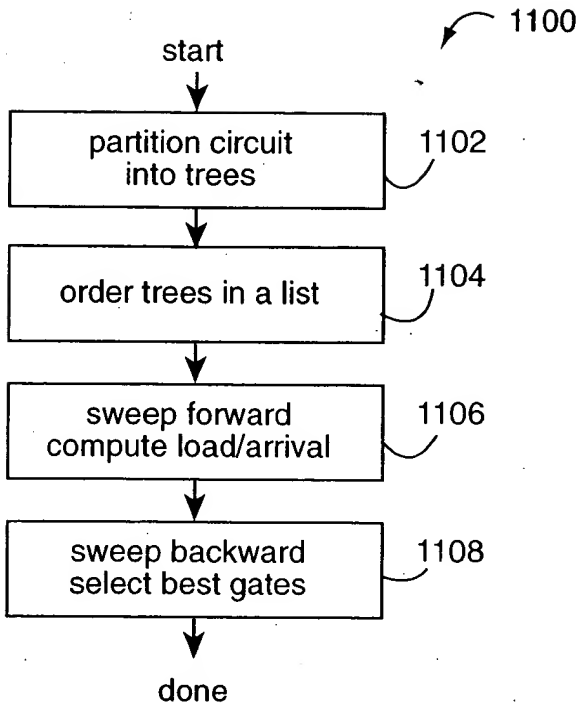

1100
start
partition circuit into trees
1102
order trees in a list
1104
sweep forward compute load/arrival
1106
sweep backward select best gates
1108
done

Fig. 12

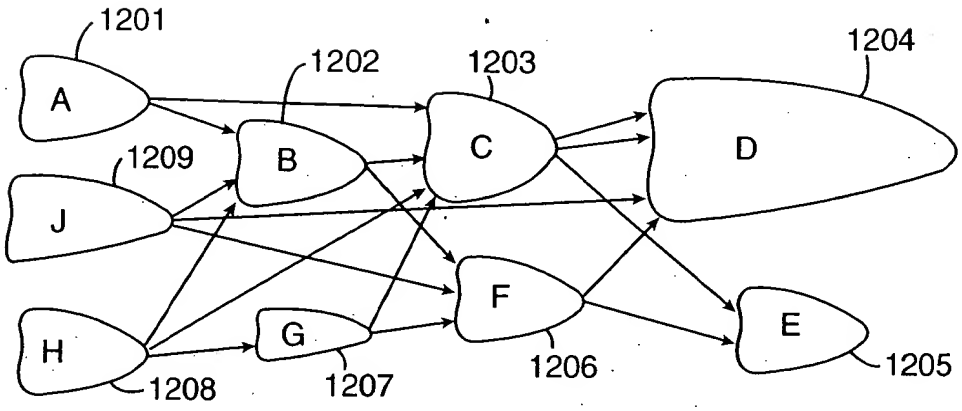

1201
1202
1203
1204
A
B
C
D
1209
J
F
H
G
E
1208
1207
1206
1205

Fig. 13

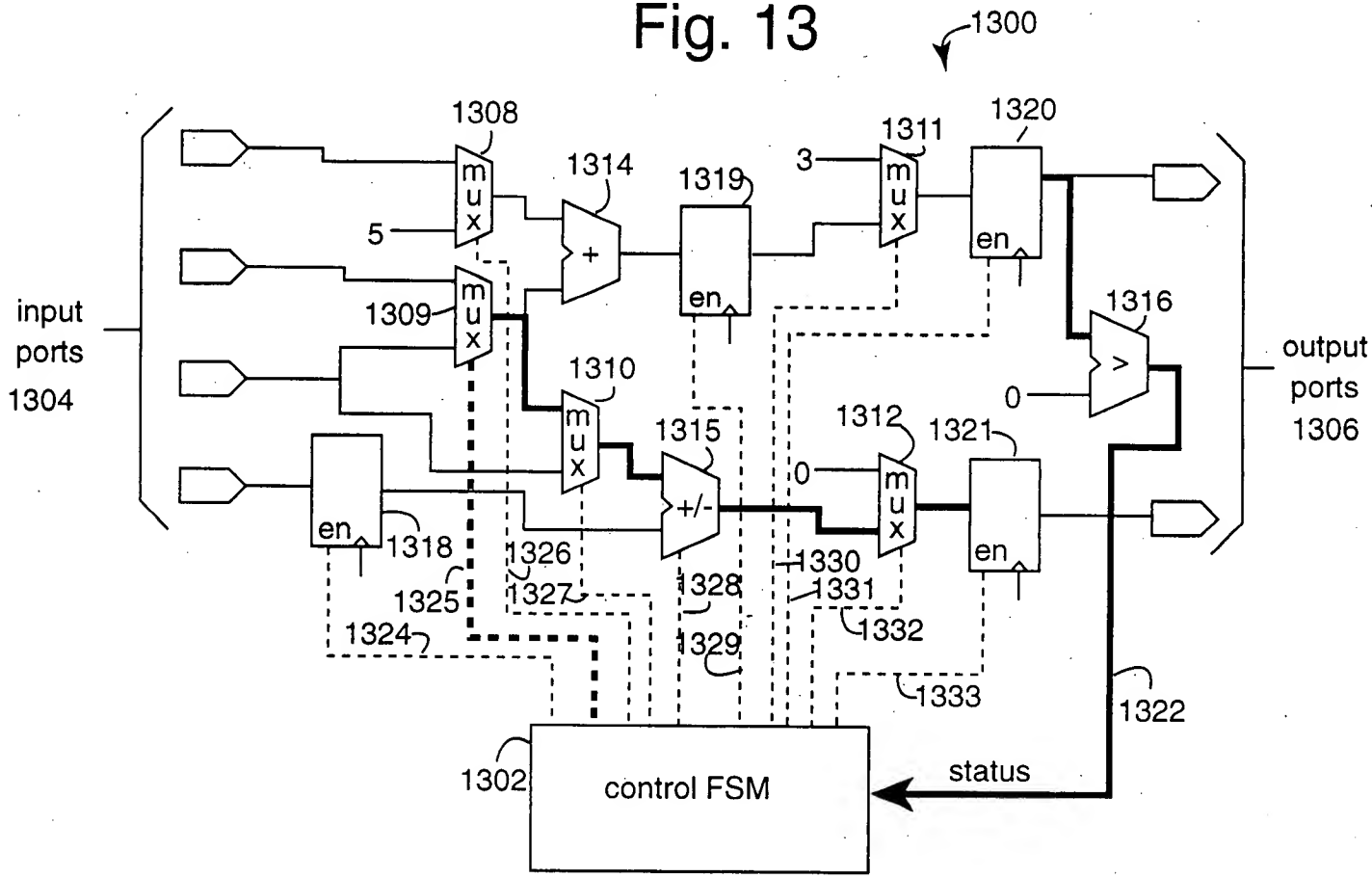


Fig. 14

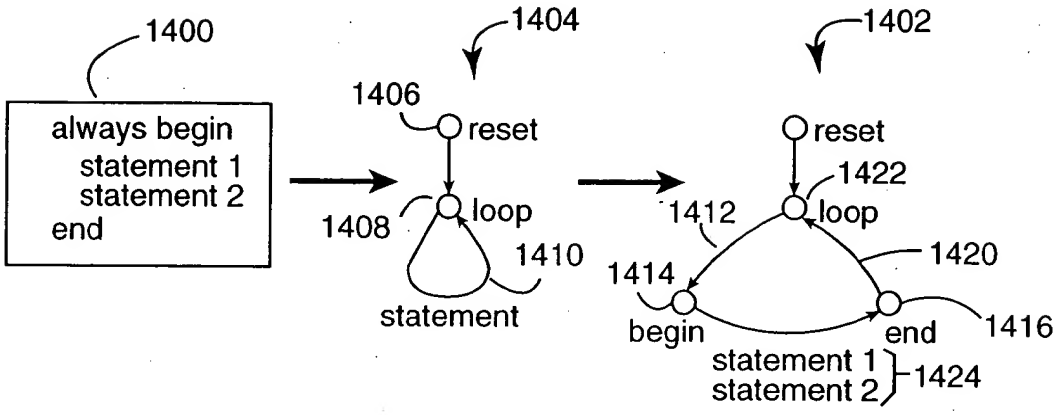

1400
always begin
statement 1
statement 2
end
1404
1406
reset
1408
loop
1410
statement
1402
reset
1422
1412
loop
1420
1414
begin
end
1416
statement 1
statement 2
1424

Fig. 15

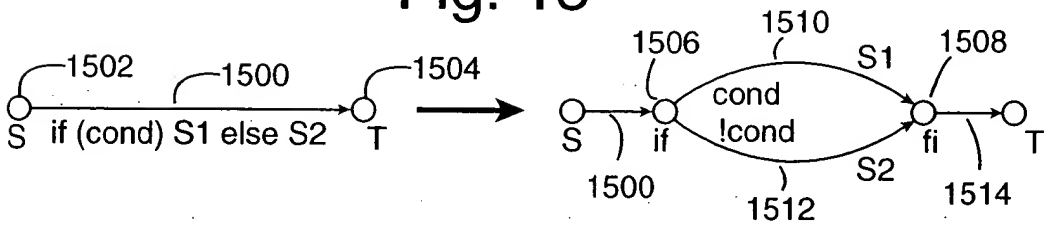

1502
1500
1504
S
if (cond) S1 else S2
T
1506
1510
S1
1508
cond
!cond
S
if
fi
T
S2
1500
1512
1514

Fig. 16A

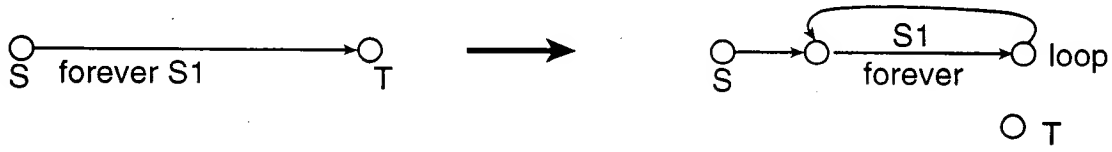


Fig. 16B

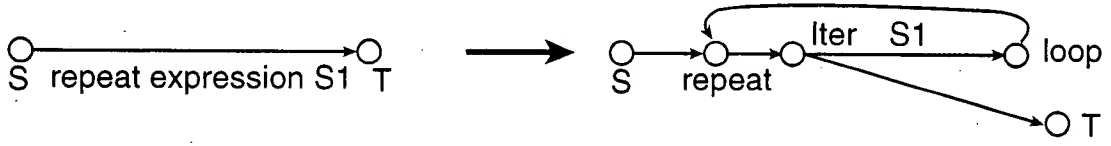


Fig. 16C

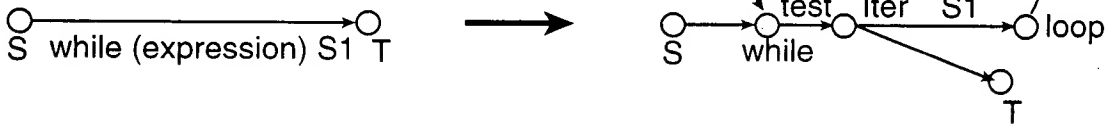


Fig. 16D

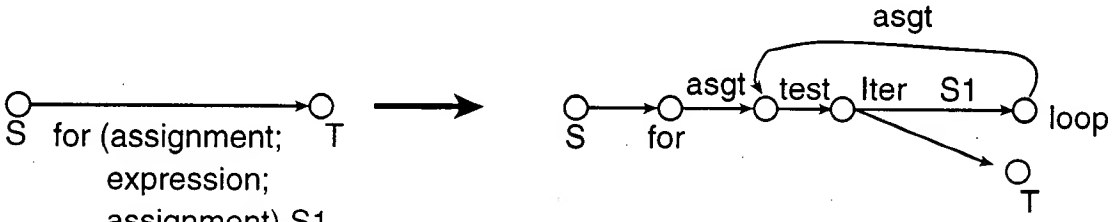


Fig. 17

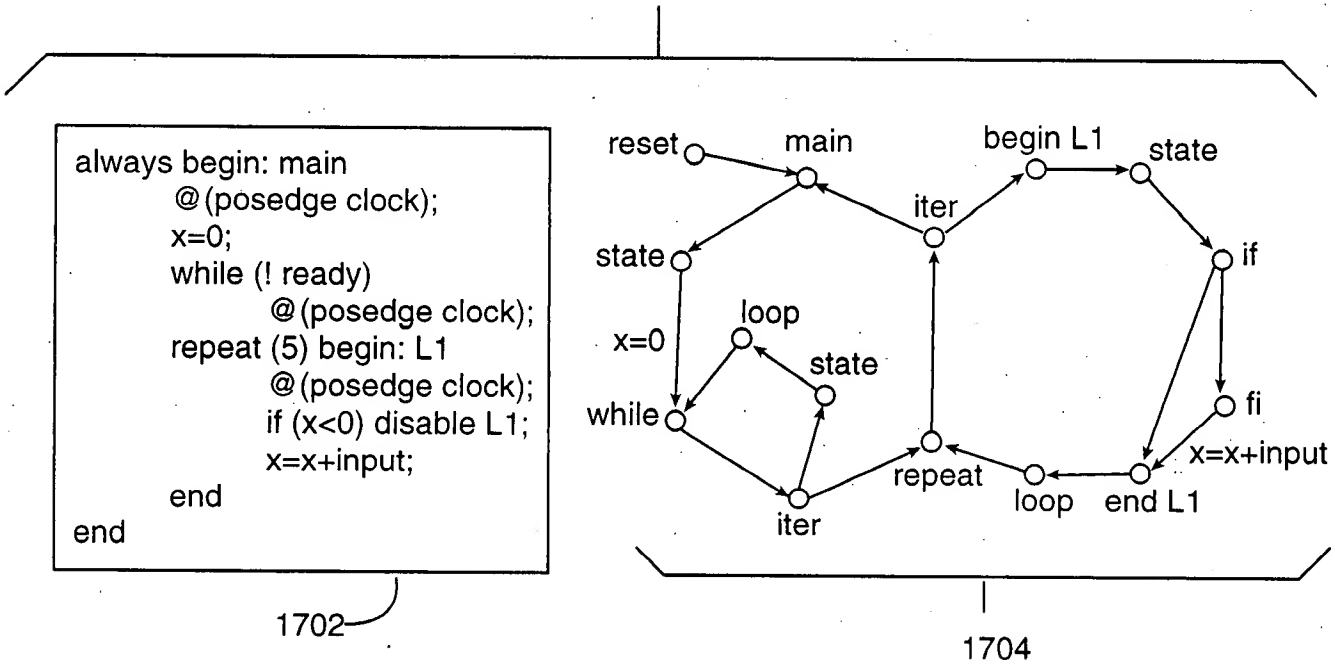


Fig. 18

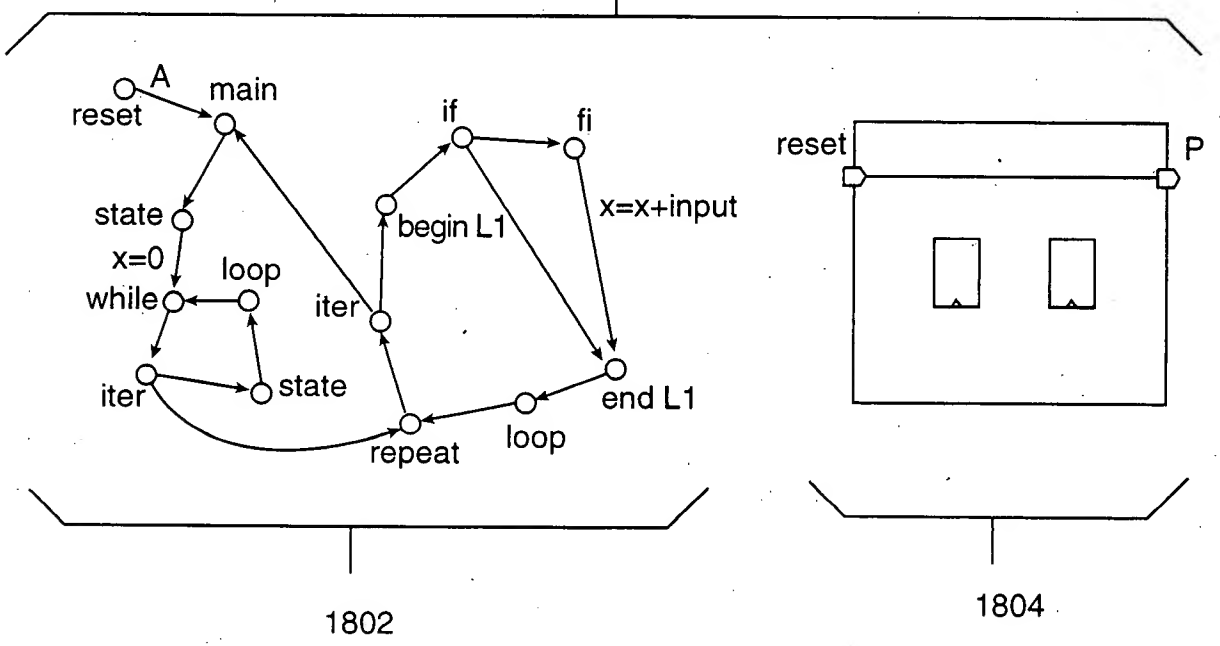


Fig. 19A

Fig. 19B

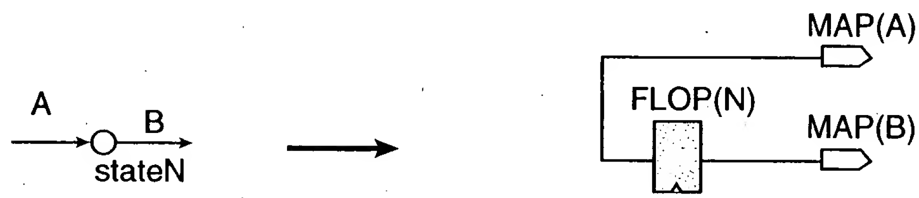


Fig. 19C

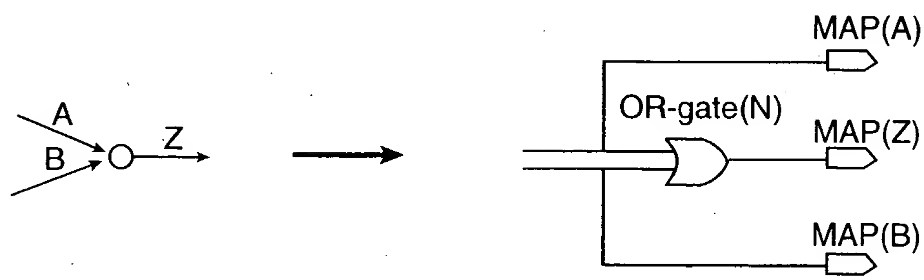


Fig. 19D

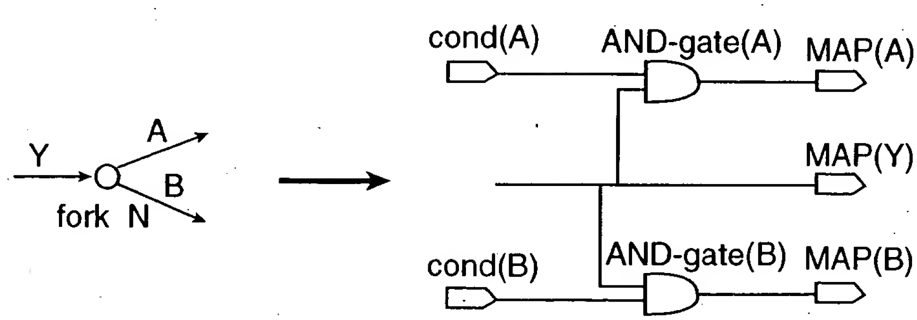
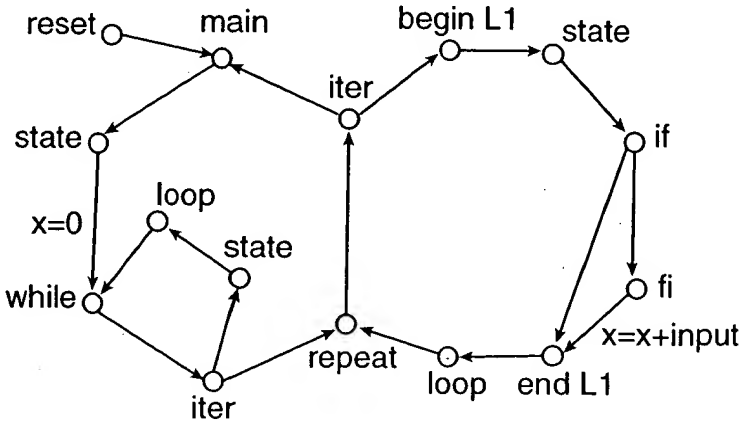


Fig. 20

```
always begin: main
        @(posedge clock);
        x=0;
        while (! ready)
                @(posedge clock);
        repeat (5) begin: L1
                @(posedge clock);
                if (x<0) disable L1;
                x=x+input;
        end
end
```

2002



2004